Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-module-3-number-guesser-4/grade/vvh

Course: IT114-003-F2024
Assigment: [IT114] Module 3 Number Guesser 4
Student: Valeria C. (vvh)

Submissions:

Submission Selection

1 Submission [submitted] 9/30/2024 4:40:05 PM ▾

Instructions

⌃ COLLAPSE ⌃

Overview Video: https://youtu.be/ej6IWrg9XjE

1. Create the below branch name
2. Implement the NumberGuess4 example from the lesson/slides
    1. https://gist.github.com/MattToegel/aced06400c812f13ad030db9518b399f
    2. Add/commit the files as-is from the lesson material (this is the base template).
    3. Push the changes to the HW branch and create a pull request to keep open until this assignment is done
3. Pick two (2) of the following options to implement
    1. Display higher or lower as a hint after a wrong guess (only after a wrong guess that doesn't roll back the level)
    2. Implement anti-data tampering of the save file data (reject user direct edits)
    3. Add a difficulty selector that adjusts the max strikes per level (i.e., "easy" 10 strikes, "medium" 5 strikes, "hard" 3 strikes)
    4. Display a cold, warm, hot indicator based on how close to the correct value the guess is (example, 10 numbers away is cold, 5 numbers away is warm, 2 numbers away is hot; adjust these per your preference) Only display this when the wrong guess doesn't roll back the level
    5. Add a hint command that can be used once per level and only after 2 strikes have been used that reduces the range around the correct number (i.e., number is 5 and range is initially 1-15, new range could be 3-8 as a hint)
    6. Implement separate save files based on a "What's your name?" prompt at the start of the game (each person gets their own save file based on user's name)
4. Fill in the below deliverables
5. Save changes and export PDF

6. Git add/commit/push your changes to the HW branch
7. Create a pull request to main (if not done so before)
8. Complete the pull request (don't forget to locally checkout main and pull changes to prep for future work)
9. Upload the same PDF to Canvas

Branch name: M3-NumberGuesser-4

Group

100%

Group: Implementation 1
Tasks: 1
Points: 4

COLLAPSE

Task

100%

Group: Implementation 1
Task #1: Implementation Evidence
Weight: ~100%
Points: ~4.00

COLLAPSE

Details:
Code screenshots must have ucid/date shown as a comment in the code.

Explanations must be your own words describing the logic and how the solution code solves the problem.

Columns: 3

Sub-Task

100%

Group: Implementation 1
Task #1: Implementation Evidence
Sub Task #1: Mention which

Task Response Prompt

Explain the logic of how you solved/implemented the chosen option (concrete details). Explain how the code works, don't just paste code snippets

Response:

Sub-Task

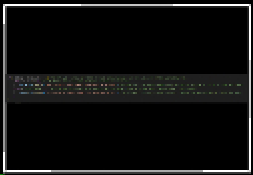
100%

Group: Implementation 1
Task #1: Implementation Evidence
Sub Task #2: Add screenshots of the

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Sub-Task

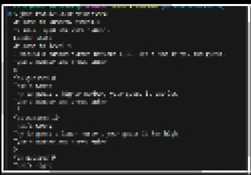
100%

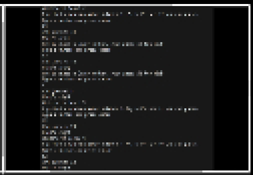
Group: Implementation 1
Task #1: Implementation Evidence
Sub Task #3: Show implementation

Task Screenshots

Gallery Style: 2 Columns

4 2 1





Item 1: Display higher or lower as a hint after a wrong guess (only after a wrong guess that doesn't roll back the level) the goal for this problem was that after the input was entered, if it was a wrong guess, give hints of higher and lower number should be guessed and also only give the hint if the wrong guess doesnt cause to lose the level.

To solve the problem of giving hints, I first needed to check if the player still has chances left. So, I implemented the condition if (strikes < maxStrikes - 1). This makes sure that the program only gives a hint if the player is not about to lose. If the strikes are less than maxStrikes - 1, it means the player still has at least one more guess before they lose, so the program can give a hint. Inside this condition, I then check if the player's guess is higher or lower than the correct number. If the guess is higher, I make the program print guess a lower number, and if the guess is lower, it would print guess a higher number

Option 1 implemented in template code Screenshot

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

first part of running program

second part of running program going to 2 levels up

part 3 after going through some levels and testing to fail one to see if it goes back to one level

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

End of Task 1

End of Group: Implementation 1
Task Status: 1/1

Group

100%

Group: Implementation 2
Tasks: 1
Points: 4

COLLAPSE

Task

100%

Group: Implementation 2
Task #1: Implementation Evidence
Weight: ~100%
Points: ~4.00

COLLAPSE

Details:
Code screenshots must have ucid/date shown as a comment in the code.

Explanations must be your own words describing the logic and how the solution code solves the problem.

Columns: 3

Sub-Task 100% Group: Implementation 2 Task #1: Implementation Evidence Sub Task #1: Mention which

Sub-Task 100% Group: Implementation 2 Task #1: Implementation Evidence Sub Task #2: Add screenshots of the

Sub-Task 100% Group: Implementation 2 Task #1: Implementation Evidence Sub Task #3: Show implementation

Task Response Prompt

Explain the logic of how you solved/implemented the chosen option (concrete details). Explain how the code works, don't just paste code snippets

Response:

Item 3: Add a difficulty selector that adjusts the max strikes per level (i.e., "easy" 10 strikes, "medium" 5 strikes, "hard" 3 strikes).

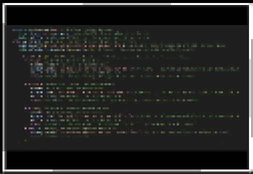
To add difficulty levels, I first implemented a method called selectDifficulty(). In this method, I ask the player to choose between easy, medium, or hard. Based on what they choose, I set a different number of strikes for the game. For example, if they choose easy, I make the program set maxStrikes = 10, so the player has 10 chances to guess before they lose. If they choose medium, I set maxStrikes = 5, and for hard, I set maxStrikes = 3 which was part of the goal of the assignment

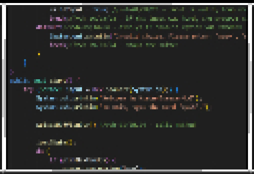
To make sure the program only accepts valid inputs like easy, medium, or hard, so I used a while loop. I implemented the condition

Task Screenshots

Gallery Style: 2 Columns

4 2 1


main method implemented on template Screenshot

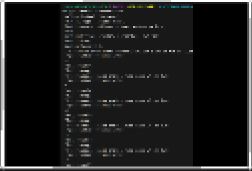

calls method screenshot

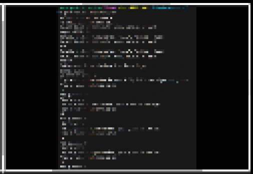
Caption(s) (required) ✓
Caption Hint: *Describe/highlight what's being shown*

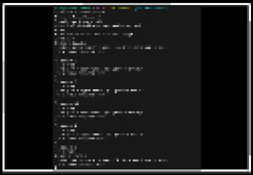
Task Screenshots

Gallery Style: 2 Columns

4 2 1


easy mode chose screenshot


testing if loop work when the wrong input is entered screenshot


medium mode selected screenshot

Caption(s) (required) ✓
Caption Hint: *Describe/highlight what's being shown*

while (!validInput) so that the program keeps asking the player for input until they type one of the valid choices. Inside the loop, I use a switch statement to check what the player typed. If they type easy, medium, or hard, I set validInput = true so the program knows the input is correct and can stop asking. If the player types something invalid, the program prints a message telling them to try again and keeps looping until they enter a valid difficulty level.

End of Task 1

End of Group: Implementation 2
Task Status: 1/1

Group

100%

Group: Misc
Tasks: 3
Points: 2

COLLAPSE

Task

100%

Group: Misc
Task #1: Reflection
Weight: ~33%
Points: ~0.67

COLLAPSE

Sub-Task

100%

Group: Misc
Task #1: Reflection
Sub Task #1: Learn anything new? Face any challenges? How did you overcome any issues?

Task Response Prompt

Provide at least a few logical sentences

Response:

I like this assignment because we get to pick what problems we would like to solve and I felt comfortable with the 2 I choose because I am still learning. I would say I had to go over switch statements for the topic 3 I chose so I did not overwhelm myself making if and else statements which helped a lot. This assignment helped me a lot in terms of seeing how the program and the logic of it works through all of it.

End of Task 1

Task

100%

Group: Misc
Task #2: Pull Request URL
Weight: ~33%
Points: ~0.67

^ COLLAPSE ^

ⓘDetails:
URL should end with /pull/# where the # is the actual pull request number.

Task URLs

URL #1
https://github.com/vvh24/vvh-IT114-003/pull/7

URL
https://github.com/vvh24/vvh-IT114-003/pull/7

End of Task 2

Task

100%

Group: Misc
Task #3: Waka Time (or related) Screenshot
Weight: ~33%
Points: ~0.67

^ COLLAPSE ^

Checklist

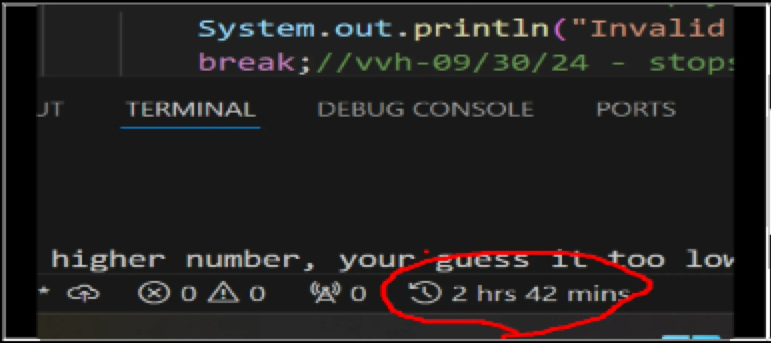
*The checkboxes are for your own tracking

#	Details
□ #1	Screenshot clearly shows what files/project were being worked on (the duration of time doesn't correlated with the grade for this item)

Task Screenshots

Gallery Style: 2 Columns

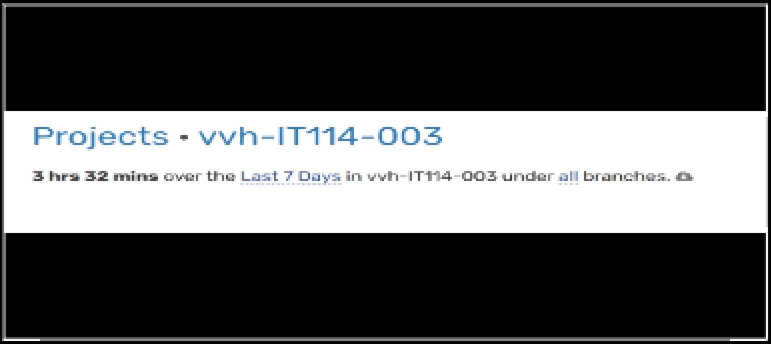
4 2 1


```
        System.out.println("Invalid
        break;//vvh-09/30/24 - stop:
```

```
higher number, your guess it too low
```

wakatime clock displaying in vscode

Wakatime tracker for each file



Projects • vvh-IT114-003

3 hrs 32 mins over the Last 7 Days in vvh-IT114-003 under all branches.

time spent in vscode

End of Task 3

End of Group: Misc
Task Status: 3/3

End of Assignment